Project-Team PASS

Processes for Adaptive Systems

Brest

Activity Report

2014

1 Team

Team leader
Antoine Beugnard, Professor, Telecom Bretagne

Administrative assistant
Armelle Lannuzel, Telecom Bretagne

Telecom Bretagne staff
Fabien Dagnat, Associate Professor
Julien Mallet, Associate Professor
Siegfried Rouvrais, Associate Professor
Maria-Teresa Segarra, Associate Professor

PhD students
Iyass Alloush, since October 2011
Anthony Lee, Cifre Orange Labs, since January 2011
Sébastien Martinez, Région Bretagne grant, since September 2012
Ngoc Tho Huynh, Vietnam grant, since December 2013

Associate members
Jean-Marie Gilliot, Associate Professor, Telecom Bretagne
Sylvain Guérin, Software Engineer, Openflexo
Christophe Guychard, Software Engineer, Openflexo
Claire Lassudrie, Associate Professor, Telecom Bretagne
Vincent Leildé, Software Engineer, Openflexo

2 Overall Objectives

2.1 Overview

Adaptive software aims at adjusting various artifacts or attributes in response to changes in the system operating environment. By operating environment we mean everything that may affect the software system behavior and its properties, for example changing requirements, execution platform, and resources. Adjustment can be performed at design time and leads to the disruption of software operation. However, the higher demand for ubiquitous, pervasive, embedded, and mobile environments, has led to the need of runtime adaptation, i.e., without software disruption. Runtime adaptation increases software complexity as the software itself has to cope with variability. Complexity is even increased if the software system is distributed. Indeed, adjusting operations may involve artifacts running on different execution platforms and adaptation mechanisms available on them may vary.

On the other hand, software complexity is traditionally tackled by focusing on the software product itself or its related design process. The first one mainly proposes artifacts such as languages, software architectures, frameworks, and middleware that help on mastering complexity at a manageable degree for designers and maintainers. The second one is interested in providing a methodology (process) to design a software system that satisfies users requirements. It guides developers in taking into account important functional and non functional concerns and may be reused for each development. Both areas offer different but interesting approaches that help in managing software complexity.

The PASS team aims at combining both approaches to master adaptive software complexity. More specifically, it aims at defining system and process models and their relationships in order to ease the development and the adaptability of software systems. The mottos of the PASS research are:

1. *Adaptation requires to reify the software design process and trace its enactment.*

   In order to build adaptive software, it is necessary to reify its design process, rationales behind it, and to consider adaptivity as a first class property. Simply knowing the current variant of the software and the targeted one, as in traditional reflection-based approaches or more recently in DSPLs (Dynamic Software Product Lines), is not enough. Indeed, they do not consider relevant information for adaptation such as when it is possible to change, what can be changed, and how building blocs state should be managed.

2. *Processes are software.*

   The description, specification, and design of processes should follow the same process as software [Ost87]. As a consequence, current standard process models such as BPEL or BPMN are inappropriate since, they poorly distinguish specification from design and enactment.

3. *Components have distributed access points.*

   Classical software component models are bound to their physical execution targets (devices) and are connected through connectors (bindings) that ensure the assembly. We propose a paradigm shift releasing the software component model dependency from its underlying infrastructure. As a consequence, the access points of a single component can be distributed among various devices and (remote) communications are managed inside components.

Our research is intended to be *vertical* in the sense that all aspects of systems are of interest: specification, design and implementation.

Solutions to enact these mottos are at the intersection of model driven engineering, software architecture, process modeling, verification and validation, and distributed systems. We intend to scientifically contribute to the three first domains (model driven engineering, software architecture, process modeling).

[Ost87] L. Osterweil, "Software processes are software too", *in: Proceedings of the 9th international conference on Software Engineering*, IEEE Computer Society Press, p. 2–13, 1987.

In this scientific context, we are interested in three kinds of systems: large scale, highly dynamic (mobile, P2P) and strongly constrained (real-time, embedded) where non functional properties such as safety, performance and reliability are important. For instance, ambient assisted living, satellites, cloud, mobile games, pervasive learning, etc.

On the application side, the PASS team explores the application to software systems such as satellite, home-automation, multi-users games, etc but also to non-software systems such as higher educational systems.

2.2 Key Issues

In its current state, PASS studies the following key issues:

- Adaptation decision and execution require process information to trigger and execute adaptations.

  What is the information to be traced? How the process can keep track of this information?

- Non-functional properties (e.g., security, performance, dependability, scalability) and other quality attributes can be considered as drivers for software and enterprise architecture design.

  In 2014, we started a focus on localization. Dealing with the placement of pieces of software is almost always relegated to the configuration and deployment stage. We believe that localization issues have to be considered earlier in order to propose dedicated design process and tools for verification and automatic deployment and adaptation.

  How methods, tools, models and processes could support non-functional (and especially localization) specifications for analysis?

- Many software component models have been developed over time. They all rely on an explicit binding with connector.

  Can't we provide an efficient component model that hides connectors?

- Software deployment relies on packet distribution with coarse-grain updating operators.

  Are there other levels of deployment? Are there other units of adaptation? What are the proper control primitives for updating software?

- Educational Systems are manageable from a systemic perspective. Models may provide a framework for strategic planning, quality enhancement, and be the pillars of renewal processes to meet the European 2020 strategic agenda for higher education.

  How to qualify and quantify the quality attributes of educational systems? How to architect educational systems and with which models? What kind of adaptation are required? How to continuously enhance and flexibly renew such systems?

3 Scientific Foundations

3.1 Modeling and Meta-Modeling

Modeling is a central activity of science and engineering. Computer science brought the idea to think about models as objects of interest. The reification of models leads to study the ways models are described. In the context of software design, with the importance of the UML modeling approach, the Object Management Groups made the Model Driven Architecture (MDA) initiative. This approach is generalized to system modeling and system of system modeling. Earlier, computer scientists had defined a theory of languages and grammar, that can be compared to the model and meta-model approach of the OMG. In fact, both approaches are very close; formalisms, theory change but underlying principles are close, if not identical.

In [Mor67], W. Morris has shown that learning of models is not learning of modeling and had made suggestions to enhance the process of developing the intuitive skill associated with model building. Almost 50 years later, the understanding of models has improved thanks to its formalization and the development of tools. However, current approaches (UML, DSL, MOF, EMF, XML, OWL, etc.) have shown that none is sufficient and that users (scientists, engineers, developers, etc.) need *flexibility*, *merging* and *preciseness*. Flexibility, since existing meta-models are not offering all required concepts; as an example, UML proposes a partial solution with its profiles. Merging, since different points of view need to be combined; the proliferation of architecture framework[Web14] is another example. And Preciseness, since the interest of tools is not only drawing or writing, but its also checking, proving, testing, generating code or documentation with a controlled semantics.

Glossary :

Meta-model A set of rules that a model has to conform to. The definition of the set of rules can be words, drawings, formal languages, etc. Some descriptions are more easily computed than others.

Model A description of a System under Study. The description can be done with words, drawings, formal languages, programs, etc. Some descriptions are more easily computed than others.

3.2 Process Modeling

The description of processes is central to Software Engineering. From early life-cycle that were coarse grain descriptions to fine-grain description like BPMN, the reification of activities, of who is doing something and of what is produced, has always been an issue in (software) development.

The current process modeling languages appear to have two distinctive problems. First, they seem to ignore the importance of a consistent approach that handles process in all the stages of its lifecycle. Either a single process model seems to represent a process in all phases (e.g. at specification phase and the implementation phase) or it has to be transformed to a completely different approach for enactment. For example BPMN models processes in all

[Mor67] W. T. Morris, "On the art of modeling", *Management Science 13*, 12, 1967, p. B707–B717.

[Web14] I. . Website, "Survey of Architecture Frameworks", `http://www.iso-architecture.org/42010/afs/frameworks-table.html`, visited 31/1/2014.

stages of development through a single notation, however it does not offer the possibility to enact them. Consequently, process developers are bound to transform the models to BPEL for enactment. Second, most of the approaches focus on the flow of activities defining the order of (presumable) execution. Some approaches use event based mechanisms to induce reactivity, but still they their focus remains on the flow of activities. Approaches like Event-driven Process Chains (EPCs) use both types of inputs for the activities (events and artifacts) together, which clutters the process model. For instance, mixing at the same time data flow and control flow in a process, makes it hard to conceptualize the interactions of an activity with its context.

Software processes just like software systems are based on the notion of lifecycle, where each stage of development has different concepts to frame. Each phase of a software process organizes different factors and issues related to the degree of its maturity in terms of completeness. Fuggetta defined a software process as, the coherent set of policies, organizational structures, technologies, procedures, and artifacts that are needed to conceive, develop, deploy, and maintain a software product. But we tend to think more like the Osterweil's analogy of a software process, where he advocates that processes share the same nature and complexity as the software system and should be treated the same way [Ost87]. Thus we are of the view that software processes themselves need to be conceived (specification), developed (implementation), deployed (enactment) and maintained (monitoring). As the life-cycle phase of a software process advances, the focus on issues related to the process also changes. For example, in specification phase the focus remains on which artifacts would be handed over by the activity. On the other hand, for enacting a process the focus shifts to when should an artifact be handed over by this activity. During the enactment phase, the reasoning about the activity is not targeted towards the choice of its inputs and outputs, rather its directed towards the related hows and whens.

Glossary :

Activity A time period dedicated to the production of Artifacts by persons playing their Roles in the Activity. An activity - as a program - has a specification, several implementations resulting from design choices, and many enactments.

Artifact Any human-made product resulting from an Activity. Documentation, source code, review, test plan are some examples of artifacts.

Process An organized set of activities, roles and artifacts.

Role An abstraction of a position denoting a responsibility in an Activity.

3.3 Software Components

Software components is an old idea. The first reference is usually attributed to M.D. McIlroy [McI69] in 1968. Since then, many component models were proposed and many surveys compare them. Components may be classified according to their lifecycle, to their interface specification, to their computational model, their implementation targets and many other dimensions.

Software components are units of composition, units of specification, units of management.

[Ost87] L. Osterweil, "Software processes are software too", *in : Proceedings of the 9th international conference on Software Engineering*, IEEE Computer Society Press, p. 2–13, 1987.

[McI69] M. McIlroy, "Mass produced software components", *Software Engineering Concepts and Techniques*, 1969.

As such they have their own development lifecycle/process. In [CSVC11], Crnkovic *et al* classify more than 20 components models with respect to their component lifecycle. It is worth noticing that these models do not propose any progression in their description. A component model is usually described at a selected level - often implementation, sometimes specification - with no references to its various forms from high level specification to low level implementation.

Software components as units are defined by their interface. It is now well admitted that a good way to describe these interfaces is through contracts [3]. However, if low level contracts (syntactic and semantics) are well established, higher levels (behavior and QoS) are seldom used.

Glossary :

Abstraction a mechanism and practice to reduce and factor out details so that one can focus on few concepts at a time.

Component a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties.

Composition a system design principle that deals with the inter-relationships of components. A highly composable system provides recombinant components that can be selected and assembled in various combinations to satisfy specific user requirements.

3.4 Adaptation and reconfiguration

Adaptation can be understood in, at least, two ways. First, as a problem of plugging two (or more) parts together, even if they were not initially designed for that. Second, as the problem of changing something (a computation, a structure) to something else (another computation or another structure). Both are not totally independent. In his survey[Kel08], S. Kell proposed a large transdisciplinary survey of software adaptation techniques (first meaning). He discussed three principles that are advocated: component models, first-class connection and loose coupling. It is worth noticing this survey covers a spectrum close to the one presented in this report. For the second interpretation, the first abstract proposal comes from autonomic computing and is modeled by the MAPE-K loop[map05]. This model states that adaptation requires to share Knowledge and to process a loop of four activities: Monitor (the system and its environment), Analyze (compute and decide to trigger an adaptation), Plan (compute how to change) and Execute (actually do the changes). This model is largely adopted.

In a distributed component-based software, a set of components located in different nodes, cooperate in order to provide services. Making such a system adaptive requires including adaptation mechanisms that are themselves distributed. Several works exist that propose distributed algorithms for either deciding or planning changes [FDP+12]. Although those ap-

[CSVC11] I. Crnkovic, S. Sentilles, A. Vulgarakis, M. Chaudron, "A classification framework for software component models", *IEEE Transactions on Software Engineering 37*, 5, 2011, p. 593–615.

[Kel08] S. Kell, "A Survey of Practical Software Adaptation Techniques", *Journal of Universal Computer Science 14*, 13, 2008, p. 2110–2157.

[map05] "An Architectural Blueprint for Autonomic Computing", *research report*, IBM, June 2005.

[FDP+12] F. Fouquet, E. Daubert, N. Plouzeau, O. Barais, J. Bourcier, J.-M. Jézéquel, "Dissemination of reconfiguration policies on mesh networks", *in : DAIS 2012*, Stockholm, Sweden, June

proaches contribute to simplify the task of building an adaptive software, they fail to provide an architectural view of the adaptation mechanisms. Such a view would allow designers to choose different distribution strategies according to their needs, i.e., some components may be managed in a distributed manner while others may not.

Glossary :

Adaptive software a software that has the ability to adapt at runtime to handle such things as changing user needs, system intrusions or faults, changing operational environment, and resource variability.

Dynamic or runtime adaptation a change on a software that takes place after deploying it. It includes deciding about the change to perform, planing the necessary operations on the software, and executing them.

MAPE-K Adaptation process loop model consisting in Monitoring, Analyzing, Planing and Executing around a shared Knowledge.

4 Application Domains

4.1 Component Software Adaptation

Participants: Antoine Beugnard, Maria-Teresa Segarra, Julien Mallet, Fabien Dagnat, Anthony Lee, Sébastien Martinez, Ngoc Tho Huynh.

Apply the mottos 1, 2 and 3.

Domains : Home-automation, HPC, Large scale applications.

In this research area we are interested in providing developers with methods and tools to easily build (distributed) adaptive software. To achieve this goal, we are currently working on the definition of 1) a generic, customizable model for distributed dynamic adaptation mechanisms at the middleware level, and 2) a set of tools that may be used by adaptation designers in order to ease the customization of our model for their needs.

Models for distributed dynamic adaptation mechanisms. To tackle current limitations we have been working on a model for distributed and coordinated dynamic adaptation [17]. We have proposed a model that constrains designers considering what are the dynamic adaptation services they need for their application and how many instances (and location) of the selected services should be created. According to the number of instances of the different services, coordination components may be mandatory and different coordination strategies are available to be used by designers, whether for decision, planning or execution services. In order to validate this work, we considered the distributed data management application domain and we intend to dynamically change data replication policies depending on the execution environment.

Development process for adaptive distributed software. The second research area is related to the automatic generation of dynamic adaptation mechanisms: decision making,

2012, `http://hal.inria.fr/hal-00688707`.

planning, and execution. This work is based on the main result of Eric Cariou and Chantal Kaboré PhD work, the Cloud Component approach [8][12], a communication abstraction (Cloud Component) used to model distributed services and a refinement process that allows for automatic generation of architectural variants. It is included in a current research trend that proposes DSPLs (Dynamic Software Product Lines) and MDE (Model-Driven Engineering) to automatically compute the actions to be performed to build the target variant when an adaptation is needed, i.e., to compute the "adaptation plan" or "reconfiguration plan". In current approaches, these actions correspond to operations on the application architecture (add component, remove component, etc...). These works propose general enough approaches that can be applied to all model-based applications, but they fail to provide a systemic approach for automatic generation of adaptation plans.

Compared to these approaches, in this research area we are interested in investigating how to generate not only architectural modification actions but also (all the) actions of an adaptation plan including application-dependent actions. We claim that during the development process relevant information may be collected then exploited to compute adaptation plans.

We have been working on one type of application-dependent actions: those related to the data managed by the components affected by an adaptation. Indeed, when a component is modified or replaced its data should also taken into account. These data are ignored by most of current approaches that assume stateless components. To tackle this issue, we have been working on two areas :

- The first one aims at helping a developer to explicitly add information related to the data managed by the application [13]. This information is related to the entities that manage data and the operations used to managed them. Such information is then used to compute data-related actions to be added to adaptation plans. More specifically, data transfers between components can be computed and added to the adaptation plan.

- The second one is related to the concept of variation point in DSPLs and its reification at runtime. Indeed, current approaches assume that all variation points in a DSPL of an application should be available at runtime: each time an optional feature is present in the DSPL, the application should be able to change at runtime to include or remove the feature. In this work, we claim that each variation point identified in the DSPL may be reified at runtime and we investigate a development process that allows to reify such variation points [14].

4.2 Process Design

Participants: Antoine Beugnard, Fabien Dagnat.

Apply the mottos 1, 2.

There is an increasing trend to consider the processes of an organization as one of its highly valuable assets. Processes are the reusable assets of an organization which define the procedures of routine working for accomplishing its goals. The software industry has the potential to become one of the most internationally dispersed high-tech industry. With growing importance of software and services sector, standardization of processes is also becoming crucial

to maintain credibility in the market. The design of a software development process follows a lifecycle that is very similar to the software development lifecycle. The process has to be designed abstractly then refined until it become an executable asset. Moreover, the multiple phases of a process development lifecycle follow an iterative/incremental approach that leads to continuous process improvement. This incremental approach calls for a refinement based strategy to develop, execute and maintain software development processes.

To be fully reusable and analyzable, a process should be modeled. Current standard models like BPMN or SPEM represent processes as artifact flows among a set of activities. To fully ensure that a defined process is applied, the process should be executable. Here executable means that the process should be enacted with the help of software. There is a tension between being executable and the abstract representation of languages such as BPMN. We propose to use refinement and model oriented technology to reduce this tension. An abstract model not executable but highly reusable is designed (or reused) and then refined to a concrete executable model of the same process but fully instantiated. Furthermore, we use a concrete model relying on event rather than connected flow to be able to support the reconfiguration at runtime of a process.

4.3 Educational Systems

Participants: Siegfried Rouvrais, Claire Lassudrie, Antoine Beugnard.

Apply the mottos 1, 2.

An educational system provides services to students and society. In a global context and for strategic alignment, the management of educational transformation is of decisive importance to Higher Educational Institutions (HEI) and their regulation bodies. During the last decade, various models of quality management have emerged in the form of quality assurance standards [11](e.g., accreditation bodies, European or international frameworks) that guide educational program leaders, designers or deans of academics to evaluate and improve educational systems including syllabus, curricula, workforce, workspaces, support services, or administrative processes. However, to date, educational system complexity is traditionally tackled by focusing restrictively on the course or curricula contents. There is no standard commonly accepted for conceptually describing and evaluating the overall complexity of educational systems, restricting the modernization agenda of HEI nationally, in Europe, and internationally.

The actual complexity of educational systems and the dynamic of their quality requirements call today for sound engineering principles. The aim of this research is thus to define and provide methods, processes and tools to design and evaluate [10] educational systems that satisfy requirements with efficient resources [11]. Based on modeling, meta-modeling and business process modeling foundations, well defined models and processes for continuous improvement could stimulate a program reform, transformation or renewal to better align with requirements (among other learning outcomes). Educational frameworks can be defined, with varying degrees of rigor, to prepare and organize various views and models for stakeholders, manage processes, and systematize analysis.

From this perspective, applying the mottos 1, 2 and based on a convergence of system en-

gineering, enterprise architecture, and sound educational design principles, this PASS research track expects to provide in-depth and usable models, methods, and tools to address issues of educational system design, transformation, and adaptability.

5 Software

5.1 ReCaml

Participants: Fabien Dagnat [contact point].

To fix bugs or to enhance a software system without service disruption, one has to update it dynamically during execution. Most prior dynamic software updating techniques require that the code to be changed is not running at the time of the update. However, this restriction precludes any change to the outermost loops of servers, OS scheduling loops and recursive functions. Permitting a dynamic update to more generally manipulate the program's execution state, including the runtime stack, alleviates this restriction but increases the likelihood of type errors. ReCaml [7] is a language for writing dynamic updates to running programs that views execution state as a delimited continuation. ReCaml includes a novel feature for introspecting continuations called match cont which is sufficiently powerful to implement a variety of updating policies. We have formalized the core of ReCaml and proved it sound (using the Coq proof assistant), thus ensuring that state-manipulating updates preserve type-safe execution of the updated program. We have implemented ReCaml as an extension to the Caml byte-code interpreter and used it for several examples.

5.2 Pymoult

Participants: Sébastien Martinez [contact point], Fabien Dagnat.

When willing to update software at runtime, one choses a platform that will give mechanisms to modify the software during its execution. For each problematic of dynamic software updating, a given platform would offer one unique answer. This means that the choice of the update mechanisms to use for modifying a running application does not belong to the developer of the application or of its updates.

Pymoult [16] is a Python library that enables application and updates developer to choose the mechanisms they want to use when designing dynamically updatable applications or dynamic updates. For that purpose, Pymoult provides the mechanisms implemented in several platforms from the literature. Pymoult also proposes a generic API for designing updates independently from any platform used.

Pymoult uses a modified version of Pypy: a Python interpreter written in Python. This modified version of Pypy allows specific interpreter operations such as intercepting object creation or capturing and modifying the runtime of threads. Pymoult was tested on several test cases from other platforms, proving itself to be a good platform for testing and prototyping dynamic software updates.

Pymoult is a free software available online at `https://bitbucket.org/smartinezgd/pymoult`.

5.3 MetaModMap

Participants: Thang Pham Quyet [previous PhD], Antoine Beugnard [contact point].

MetaModMap is an eclipse-based plugin which allows to define the intensional semantics correspondences of elements between two similar metamodels. From this definition, the plugin provides a facility which rewrites a legacy transformation definition for the old metamodels to a new one for the new similar metamodels.

This software is developed using Maude programming language and its extension Moment2.

5.4 Openflexo

Participants: Antoine Beugnard, Fabien Dagnat, Christophe Guychard [contact point], Sylvain Guérin [SCIC[1] openflexo].

Openflexo is an open-source (GPL3) business architecture platform that supports collaborative agile missions by seamlessly transforming multifaceted enterprise models into business oriented deliverables. Openflexo Modeler provides frequent delivery production cycles (storyboard, documentation, operational prototypes and applications). Openflexo is business oriented. It's a tool gathering all actors around a shared visualization of a multifaceted model, continuously updated and enriched at each iteration.

6 New Results

6.1 A Refinement based methodology for software process modeling

Participants: Fahad Golra, Fabien Dagnat [1].

This thesis develops a conceptual foundation for refinement based development of software processes keeping in view the precise requirements for each individual phase of process development lifecycle. It exploits model driven engineering to present a multi-metamodel framework for the development of software processes, where each metamodel corresponds to a different phase of a process. A process undergoes a series of refinements till it is enriched with execution capabilities. Keeping in view the need to comply with the adopted standards, the architecture of process modeling approach exploits the concept of abstraction. This mechanism also caters for special circumstances where a software enterprise need to follow multiple process standards for the same project.

On the basis of the insights gained from the examination of contemporary offe- rings in this domain, the proposed process modeling framework tends to foster an architecture that is developed around the concepts of "design by contract" and "design for reuse". This allows to develop a process model that is modular in structure and guarantees the correctness of interactions between the constituent activities. Separation of concerns being the motivation, data-flow within a process is handled at a different abstraction level than the control-flow.

[1]Société Coopérative d'Intérêt Collectif: a social and solidarity economy company dedicated to cooperation. `http://www.les-scic.coop/sites/fr/les-scic/`

Conformance between these levels allows to offer a bi-layered architecture that handles the flow of data through an underlying event management system. An assessment of the capabilities of the proposed approach is provided through a comprehensive patterns-based analysis, which allows a direct comparison of its functionality with other process modeling approaches.

6.2 A Framework for dynamic reconfiguration of Python applications

Participants: Sébastien Martinez, Fabien Dagnat [8].

Implementing dynamic software update (DSU) mechanisms from the literature in Pymoult led us to a design easing their combination and configuration. While state of the art DSU platforms use a selected combination of mechanisms to enable dynamic updates, Pymoult allows developers to select the mechanisms they find best suited for each update. While it gives better control on the updates to the developer, this strategy is less restrictive on application compatibles with DSU. Pymoult was successfully used to update Django powered applications[2].

We consider the design of Pymoult to be the vanguard of a generic API for DSU that could be followed by next generation DSU platforms. Such DSU platforms could be combined to update complex applications for which a single platform would not be sufficient (*e.g.* multi-language applications).

We have applied Pymoult to propose a new component platform supporting dynamic safe reconfiguration: Pycots. The architecture of a running Pycots application can be extracted using a reflexive level. This extraction builds a Coqcots model, a formal component model defined in Coq. Once the architecture is extracted, the developer simultaneously defines its reconfiguration and proves its correctness within Coq. Once proved, the reconfiguration script is extracted from the proof using Coq extraction facilities. Then this script is glued with DSU code developed to support the updates of component behavior. Lastly, this code is uploaded to the Pycots running application to be executed. The resulting update therefore modify the application without stopping it.

6.3 Beyond strict modelling: some modelling situations

Participants: Antoine Beugnard, Fabien Dagnat, Sylvain Guérin, Christophe Guychard [7].

In our research and teaching activities, we are used to model a lot. We produce formal models for statical analysis, semi-formal models with UML for instance (see 3.1) or even simple drawings. This led us to identify and describe modeling situations [7].

We can summarize this situations in five categories:

from the concept to the instance It is the most frequently *equipped* approach. The toolkit brings its set of concepts that can be used for modeling. The meta-model do exists before all, the modeler refers to it. Some tools allow when the realization is impossible to enlarge the set of concepts and then realized. The instance does not exist without the concept.

[2]Django is a widespread Python framework for web applications.

from the instance to the concept This is the most commonly *used* approach before the passage to the previous case. We draw on a board, a piece of paper or a drawing tool. We identify the concepts then we change and configure tools to be able to derive concepts into instances.

recognition of an interpretation Intermediate approach, it allows the independent existence of concepts and instances, and finally recognize their interpretation link.

composition Independent from the interpretation link, composition applies to both instances and concepts. It allows to introduce the concept of composition that is used to represent associations, relationships, content, etc.

evolution The evolutions highlight the dynamic of construction of the two involved levels: instances and concepts. They can be independent of the interpretation or not.

This approach is equipped in the Openflexo modeling environment (see 5.4) as a Free Modeling module. This tool allows the modeler to draw independently instances or concepts, derive an instance from a concept or identify a concept from instances, and recognize an interpretation between already existing instance and concept.

6.4 Educational Systems

Participants: Claire Lassudrie, Siegfried Rouvrais [10][11].

As first results, (i) a generic model of reference and process models for higher educational institution evaluations has been formalized [10, 11] and presented to an ISO community in 2014, (ii) the Evaluation Models and Processes for Higher Educational Systems has been defined by Telecom Bretagne PASS members, in concordance with ISO SPICE 15504 and ISO 330xx and will be published in 2015 before a potential request as domain specific proposal for Higher Educational Institutions, aka an ISO Publicly Available Standard (PAS). More specifically, following a Business Process modeling notation, in line with evaluation Reference Models, a pair-wise evaluation Process Model has been set up at the end of 2014 and specified in evaluation kits.

6.5 Other results

Publications: [2, 3, 4, 5, 6, 9]

6.6 Assessment of achievements

Participants: All participants.

The results achieved by the PASS team must be compared with the key issues presented in the objective section. Not all key issues have deserved attention yet. However, a few of them have been sufficiently well explored to start and draw conclusions.

The first key issue is “What are the information to be traced? How the process can keep trace of this information?” The information to trace are mainly the information related to

the decision choices that have an impact on the component parts that are to be adapted or modified. We experiment different models where to attach those informations. Those models are close to process models or feature models; all are some kind of process abstractions.

The second key issue is "Can we provide an efficient component model that hides connectors?" We have proposed a component model that illustrates a paradigm shift. We can develop distributed application with components that encapsulate all the communications required by the distribution, but that are composed only locally. This component model should be the beginning of new experiences and developments.

7 Contracts and Grants with Industry

7.1 Projet ANR Formose

The Formose project is an industrial research project that aims to provide a formally-grounded, model-based requirements engineering method for critical complex systems, supported by an open-source environment. Formose is a 48 months project proposed by a consortium made up of 2 academic partners (LACL - project leader - and Institut Mines-Telecom) and 2 companies (THALES and ClearSy).

It is well-known that requirements engineering (RE) is critical in software and system design. Indeed, a major part of the cost of software and system development is known to be traceable to the understanding of the problem domain and requirements. Today's current industrial practices and tools are not sufficiently efficient. They mainly consist of in-house processes, lessons learnt documents, and requirements management tools (word processor macros, traceability tools, requirements database). Even if, in the last decade, much research on this topic reached maturity, recent studies have pointed out that issues remain open. The Formose project will address several challenges raised by these issues, the most crucial in the domain of RE for critical complex systems. They concern the need of taking into account the high complexity of such systems, the need of a better integration of RE with verification and validation techniques to ensure a better quality of requirements, and more generally the need of method guidance and tool support during the process of elaborating high quality requirements models. The main results of the project will be the following.

First, we will define a requirements modeling language integrating basic concepts of existing languages, such as KAOS or Tropos/i*. To this classical concepts we will add new ones to take into account the specific characteristics of critical complex systems:

- their abstract architecture will be considered by allowing requirements to be defined at different abstraction layers and verifying their consistency;
- the language will allow to specify not only non-functional requirements related to safety and performance but also specific requirements related to the presence of different operational modes and reconfigurations in such systems.

The language will use multiple views (natural language, graphical notations, formal notations) of requirements to be understandable and easily manipulated by all the stakeholders. For verification purpose, we will adopt existing and complementary formal methods, supported

by efficient tools: the B method and the timed model-checker UPPAAL. Then we will define a highly customizable process for requirements engineering of complex systems to guide the engineers in their different activities for elaborating requirements models. To support the enactment of the process we will use the OpenFlexo platform that will integrate not only the ad-hoc tools developed during the project but also the existing verification tools (Atelier B and UPPAAL model-checker). We also envision to ensure a two way exchange: OpenFlexo will be able to send proof obligations to a prover and will also be able to get the answer from the prover and interpret it to present it to engineers, using adequate representations.

Finally, all along the project, we intend to regularly assess the method and the tools using the case studies provided by the industrial partners (THALES and ClearSy). Last but not least, THALES and ClearSy engineers will follow the method and be active users of the tools to provide real world usage return on the usability of the Formose method. ClearSy will ensure tools exploitation and maintenance beyond project completion and THALES will develop the industrial version of the tools.

The dissemination of the results will be ensured by a web site, publications in national and international journals and conferences, and free availability of the platform. We also intend to communicate towards industrial communities that are concerned by critical complex systems by presenting publications in national and international industrial conferences.

7.2 Erasmus+ KA2 EU project: Quality Assurance and Enhancement Marketplace for Higher Education Institutions

Improving European education and training system quality has been set as a key target in Europe's strategy to become a smart, sustainable and inclusive economy by 2020 (Council of the European Union, 2010). These objectives are more specifically defined in the so called Modernization Agenda (EC 2011). More specifically it sets a goal to improve the quality and relevance of higher education. In this process, external evaluation and self-assessment are seen in a key role.

There have been many European projects on Quality Assurance, but they do not cover still existing problems with accreditation such as required resources, complexity, delays between the evaluation rounds, poor feedback, poor quality loop and distance from continuous education development. There is a need for more flexible evaluation models and processes with peers to reduce the inertia of heavy accreditations/evaluations in HEIs.

QAEMP-project (2014-2017) proposes a flexible and collaborative methods, processes and tools for degree program / higher education evaluation. The innovativeness of the project lies in the collaborative model of quality assurance that can complement accreditations and existing QA systems. The project promotes and strengthens the European cooperation in quality assurance while designing and piloting new kind of continuous, accessible, cooperation based model, tools and a virtual platform (the Market Place) supporting so called cross-sparring between institutions. Cross-sparring is to be understood as a process to make feedback more collaborative, concrete and objective, thanks to critical, but discreet brainstorming sessions, where strategies can be discussed, repeatedly contributing to the quality assurance with a critical external view. The Market Place serves as a tool for finding the best possible sparring partners as well as a forum for networking, sharing experiences, information and best practices.

The QAEMP project consortium consists of eight European higher education institutions: Reykjavik University, Iceland; Turku University of Applied Sciences, Finland ; Aarhus University, Denmark; Helsinki Metropolia University of Applied Sciences, Finland; Umeå University, Sweden; Telecom Bretagne, France; Aston University and Queens University Belfast, United Kingdom.

The dissemination of the results will be ensured by a web site, publications in international journals and conferences, and accessibility of the MarketPlace.

7.3 Projet Recherche Innovante Thalès

We participate with the Openflexo SCIC to the development of a modeling editor prototype.

7.4 Openflexo

We participate to the creation of the Openflexo, a social and solidarity economy company dedicated to cooperation for the development and use of the Openflexo modeling toolkit (see 5.4).

This company was created with 3 persons and develops and animates a network of 10 cooperators (other companies or individuals).

8 Other Grants and Activities

8.1 International Collaborations

- Vincent Englebert, from Université Catholique de Namur.
- Siegfried Rouvrais
  - is the French representative for CDIO collaborator's international meetings since several years (e.g. Chalmers 2014, UPC Barcelona 2014, Univ Santiago de Chili 2014). He has been elected in 2013, during the MIT/Harvard Conference, as council member for a one year term. The CDIO Initiative (acronym for Conceive – Design – Implement – Operate) is a structured educational framework for preparing the next generation of engineers, including more than 100 prestigious Universities for STEM and engineering education worldwide . The framework provides students with an education stressing engineering fundamentals set in the context of Conceiving & Designing & Implementing & Operating real-world systems and products. Dr. Rouvrais works closely and on a regular basis with the European Region.
  - is the French correspondent of the 2014-2018 Erasmus Mundus Action 2 INSPIRE Project (INternational Science Promoting Innovation and entREpreneurship) with South Africa.
  - works closely with UpCERG and LeTech research groups, respectively at Uppsala and Aalto Universities (Sweden & Finland), on STEM and engineering education renewal.
- The PhD thesis of Ngoc Tho Huynh is a grant from the Vietnam-French cooperation.

8.2 National Collaborations

- The PASS team has a contract with Région Bretagne for the funding of S. Martinez's PhD (since september 2012). This thesis is co-advised by Jérémy Buisson of ArchWare Team of the University of Vannes / IRISA.

- We have informal project with David Chemouil and Julien Brunel from Onera, Toulouse, on an Alloy-like DSL for specifying components.

9 Dissemination

9.1 Involvement in the Scientific Community

- Antoine Beugnard has been member of the programming committee of CIEL 2014. He was also member in several doctoral committees: Jean Christophe Bach on "Un ilôt formel pour les transformations de modèles qualifiables ", Olivier Finot on "Oracles du test de transformations de modèles" and Carlos Gonzáles on "Pragmatic Model Verification". He is member of the "scientific board" of the school Saint-Cyr Coetquidan.

- Siegfried Rouvrais is an application evaluator for higher educational institutions wishing to join the CDIO initiative. He is reviewer and TPC member for the following conferences: IEEE Frontiers in Education, IEEE International Conference on Learning and Teaching in Computing and Engineering, intl. CDIO Conf.

9.2 Teaching

- Antoine Beugnard teaches software engineering, modeling and object programming. He teaches these courses at License and at Master level. He is the coordinator of the brittany research master (Master Recherche en Informatique de l'Université de Rennes 1) at Telecom Bretagne.

- Fabien Dagnat is the head of the computer science speciality at Telecom Bretagne. He teaches advanced programming (programming languages, functional programming, compilation), formal modeling (for concurrency mainly) and object oriented programming to master students of Telecom Bretagne. He teaches also a formal modeling course in the core of the brittany research master.

- Julien Mallet teaches software engineering, modeling, object oriented programming and computer security at License and at Master level. He is also involved in project-based learning.

- Siegfried Rouvrais is involved in a broad range of courses relating to computer science and software engineering both at B.Sc. and M.Sc. levels, as well as for adult professionals (e.g., Requirements, Software Architecture, Programming Languages, UML, Business Process Design and Modelling, Workflow, Enterprise Information Systems). In connection with these theoretical and practical courses, he investigates problem- and

project-based learning styles aiming at favoring students' autonomy with a focus on process and reflectivity. Since 2003, as educational program designer, he has been particularly involved in integrating such models into Telecom Bretagne curricula and vocational trainings, with a clear emphasis on student's competency development, multidisciplinarity, and industry connections. Since several years, he trains faculty members from UeB and offer consultations for improving teaching and curriculum design (PjBL, Active Pedagogy, Reflectivity, Constructive Alignment, Program Design, etc.). He co-leads the TREE research Group at Institut Mines Telecom (Transdisciplinary Research in Engineering Education).

- Maria-Teresa Segarra is the head of the computer science domain at Telecom Bretagne. She teaches advanced programming paradigms (component and service oriented), object-oriented programming and design and databases at License and Master levels.

10 Bibliography

Major publications by the team in recent years

[1] L. Besnard, T. Gautier, J. Ouy, J.-P. Talpin, J.-P. Bodeveix, A. Cortier, M. Pantel, M. Strecker, G. Garcia, A.-E. Rugina, J. Buisson, F. Dagnat, "Polychronous interpretation of synoptic, a domain specific modeling language for embedded flight-software.", *Formal methods for aerospace*, 18, 2010, p. 80 – 87.

[2] A. Beugnard, A. Hassan, "A New Component Model for Highly Distributed Environements", *in : FACS 2011 : 8th International Symposium on Formal Aspects of Component Software*, Springer (editor), 2011.

[3] A. Beugnard, J.-M. Jezéquel, N. Plouzeau, D. Watkins, "Making components contract aware", *Computer 32*, 7, july 1999, p. 38 – 45.

[4] A. Beugnard, S. Sadou, "Method overloading and overriding cause distribution transparency and encapsulation flaws", *Journal of Object Technology, Special Issue OOPS Track at SAC 2006 6*, 2, february 2007, p. 31 – 45.

[5] A. Beugnard, "OO languages late-binding signature", *in : The Ninth International Workshop on Foundations of Object-Oriented languages, FOOL 9*, p. 1 – 6, 2002.

[6] J. Buisson, E. Calvacante, F. Dagnat, S. Martinez, E. Leroux, "Coqcots & Pycots: non-stopping components for safe dynamic reconfiguration", *in : CBSE 2014 : proceedings of the 17th international ACM Sigsoft symposium on Component-based software engineering*, ACM (editor), p. 85 – 90, New York, NY, USA, 2014.

[7] J. Buisson, F. Dagnat, "ReCaml: execution state as the cornerstone of reconfigurations", *in : ACM SIGPLAN: 15th International Conference on Functional Programming*, ACM (editor), p. 27 – 38, New York, NY, USA, 2010.

[8] E. Cariou, *Architecture des applications distribuées utilisant des médiums de communication*, PhD Thesis, INFO - Dépt. Informatique (Institut Télécom-Télécom Bretagne), UR1 - Université de Rennes 1, 2003.

[9] V. Chiprianov, Y. Kermarrec, S. Rouvrais, J. Simonin, "Extending Enterprise Architecture Modeling Languages for Domain Specificity and Collaboration: Application to Telecommunications Service Design", *Software and systems modeling 13*, 3, july 2014, p. 963 – 974.

[10] J.-M. Gilliot, A. P. Khac, A. Beugnard, M. T. Segarra, "L'ingénierie dirigée par les modèles pour la conception d'applications à architectures réparties adaptables", *Technique et science informatiques 30*, 1, janvier 2011.

[11] F. R. Golra, F. Dagnat, "Generation of Dynamic Process Models for Multi-metamodel Applications", *in : ICSSP 2012: International Conference on Software and System Process*, 2012.

[12] E. Kabore, *Contribution à l'automatisation d'un processus de construction d'abstractions de communication par transformations successives de modèles*, PhD Thesis, INFO - Dépt. Informatique (Institut Télécom-Télécom Bretagne), 2008.

[13] A. P. Khac, *A model-driven feature-based approach to runtime adaptation of distributed software architectures*, PhD Thesis, INFO - Dépt. Informatique (Institut Télécom-Télécom Bretagne), UR1 - Université de Rennes 1, UEB - Université Européenne de Bretagne (UEB), 2010.

[14] J.-B. Lezoray, M. T. Segarra, A. Beugnard, J.-M. Gilliot, "Modeling Dynamic Adaptations using Augmented Feature Models", *in : SAC 2013 : 28th Symposium On Applied Computing*, 2013.

[15] J. Mallet, S. Rouvrais, "Style-Based Model Transformation for Early Extrafunctional Analysis of Distributed Systems", *in : QoSA '08 : international conference on quality of software architecture, october, Karlsruhe, Germany*, S. verlag (editor), p. 55 – 70, 2008.

[16] S. Martinez, F. Dagnat, J. Buisson, "Prototyping DSU techniques using Python", *in : HotSWUP 2013 : 5th Workshop on Hot Topics in Software Upgrades*, USENIX (editor), 2013. hal-00907744.

[17] M. Zouari, M. T. Segarra, F. André, A. Thépaut, "An Architectural Model for Building Distributed Adaptation Systems", *in : 5th International Symposium on Intelligent Distributed Systems*, 2011.

Doctoral dissertations and "Habilitation" theses

[1] F. R. Golra, *A Refinement based methodology for software process modeling*, PhD Thesis, INFO - Dépt. Informatique (Institut Mines-Télécom-Télécom Bretagne-UEB), january 2014, Th. doct. : Informatique, Institut Mines-Télécom-Télécom Bretagne-UEB, january 2014.

Articles in referred journals and book chapters

[2] V. Chiprianov, Y. Kermarrec, S. Rouvrais, J. Simonin, "Extending Enterprise Architecture Modeling Languages for Domain Specificity and Collaboration: Application to Telecommunications Service Design", *Software and systems modeling 13*, 3, july 2014, p. 963 – 974.

[3] K. C. A. Lee, M. T. Segarra, S. Guelec, "A Deployment-oriented Development Process based on Context Variability Modeling", *MODELSWARD*, january 2014, p. 454 – 459.

[4] M. Zouari, M. T. Segarra, K. Drira, *Architectures logicielles et outils : gestion distribuée et coordonnée de la reconfiguration dynamique (Chap. 8)*, *Informatique et systèmes d'information : Recherche, technologie, applications*, Editions Hermes Sciences-Lavoisier, Paris, 2014, ch. Architectures logicielles : Principes, techniques et outils, sous la direction de Mourad Chabane Oussalah.

Publications in Conferences and Workshops

[5] I. Alloush, C. Aoun, Y. Kermarrec, S. Rouvrais, "A Domain-Specific Framework for Creating Early Trusted Underwater Systems Relying on Enterprise Architecture", *in : MASCOTS 2014 : 22nd International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, IEEE (editor), 2014.

[6] I. Alloush, Y. Kermarrec, S. Rouvrais, "Transforming Viewpoints of Distributed Designs to Support Simulation Scenarios", *in : ICSOFT-EA 2014 : proceedings of the 9th International Conference on Software Engineering and Applications*, p. 321 – 328, 2014. Editors : Holzinger, A.; Libourel, T.; Maciaszek, L. A. & Mellor, S. J. ; ICSOFT-EA : part of ICSOFT conference.

[7] A. Beugnard, F. Dagnat, S. Guerin, C. Guychard, "Des situations de modélisation pour évaluer les outils de modélisation", *in : INFORSID 2014 : 32ème congrès de l'INFormatique des ORganisations et Systèmes d'Information et de Décision*, p. 181 – 196, 2014.

[8] J. Buisson, E. Calvacante, F. Dagnat, S. Martinez, E. Leroux, "Coqcots & Pycots: non-stopping components for safe dynamic reconfiguration", *in : CBSE 2014 : proceedings of the 17th international ACM Sigsoft symposium on Component-based software engineering*, ACM (editor), p. 85 – 90, New York, NY, USA, 2014.

[9] M. Le Goff-Pronost, B. Vinouze, C. Lassudrie, M. Morvan, F. Gallée, D. Baux, R. Naël, M. Arzel, J.-P. Coupez, A. Beugnard, "Introducing complexity into project management through multi-stakeholder interactions", *in : SEFI 2014 : 42th annual conference*, SEFI (editor), p. 135 – 135, Bruxelles, 2014.

[10] S. Rouvrais, C. Lassudrie, "An Assessment Framework for Engineering Education Systems", *in : 14th International SPICE Conference*, S. Verlag (editor), 2014.

[11] S. Rouvrais, C. Lassudrie, "Educational program evaluations: rationalizing assessment models and processes for engineering education quality enhancement ", *in : CDIO 2014 : 10th international conference Conceive Design Implement Operate on sharing successful engineering education experiences*, Universitat Politècnica de Catalunya, 2014.